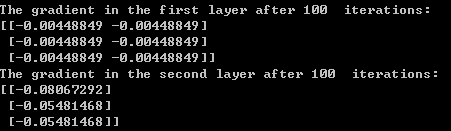
**Machine Learning HW12 Section 6100**

***Pu Zhang (661536015)***

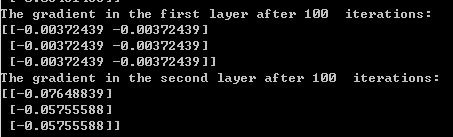
1. **Gradient Descent**
2. The gradient of with output node in the first layer is:

The gradient in the second layer is:



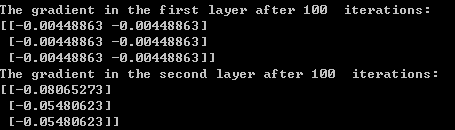
The gradient of with identity output node in the first layer is:

The gradient in the second layer is:



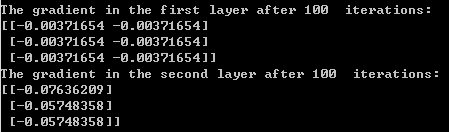
1. By randomly perturbing some weights with 0.0001, the gradient of in the first layer with output node is:

The gradient in the second layer is:



The gradient of with identity output node in the first layer is:

The gradient in the second layer is:

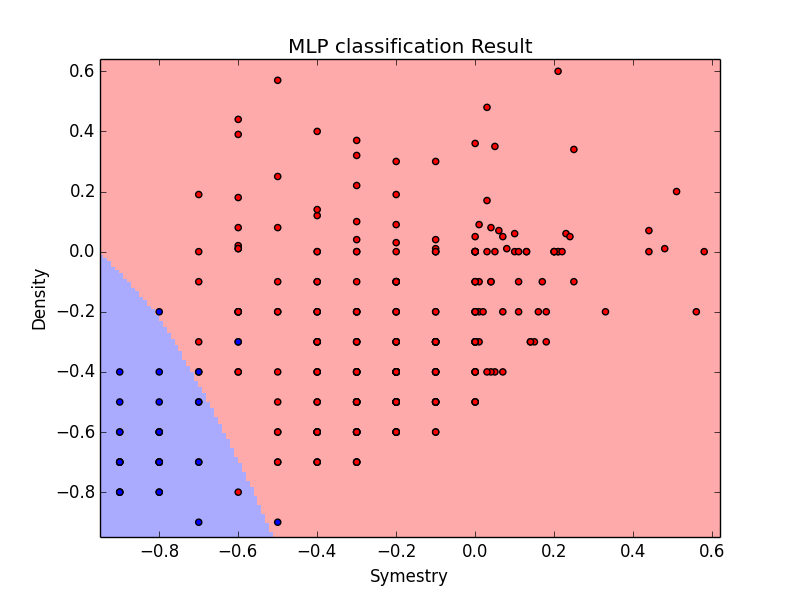


The gradient doesn’t change too much with perturbed weights compared to the original weights.

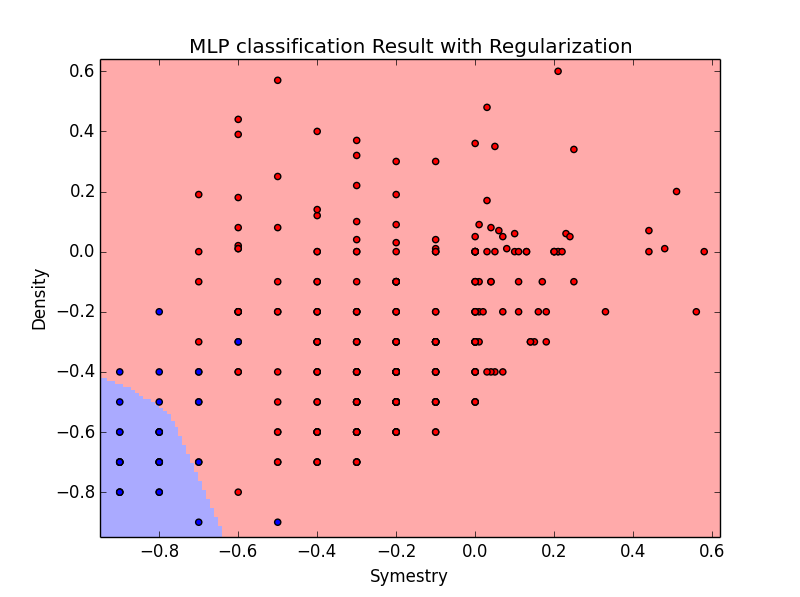
1. **Neural Network**
2. I used stochastic gradient descent instead for computational efficiency. The following figure plots v.s. iterations in SGD parameter learning:



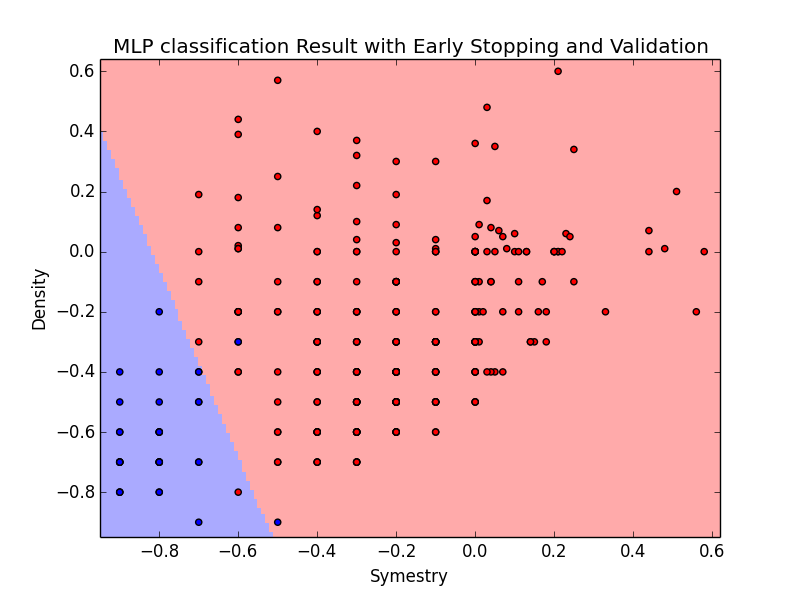
The resulting classifier is as below:



1. With the weight decay , the resulting classifier is as below:



1. With early stopping and validation, the resulting classifier is:

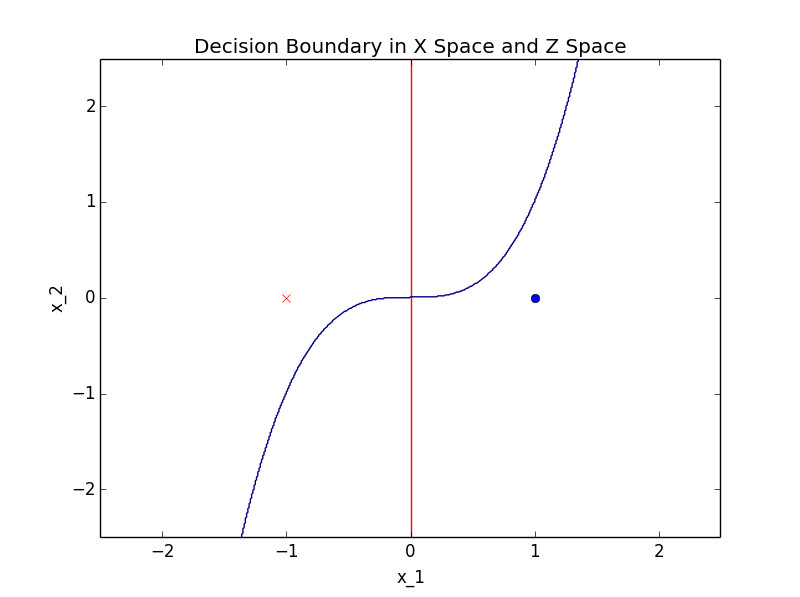




The testing error is 0.03501.

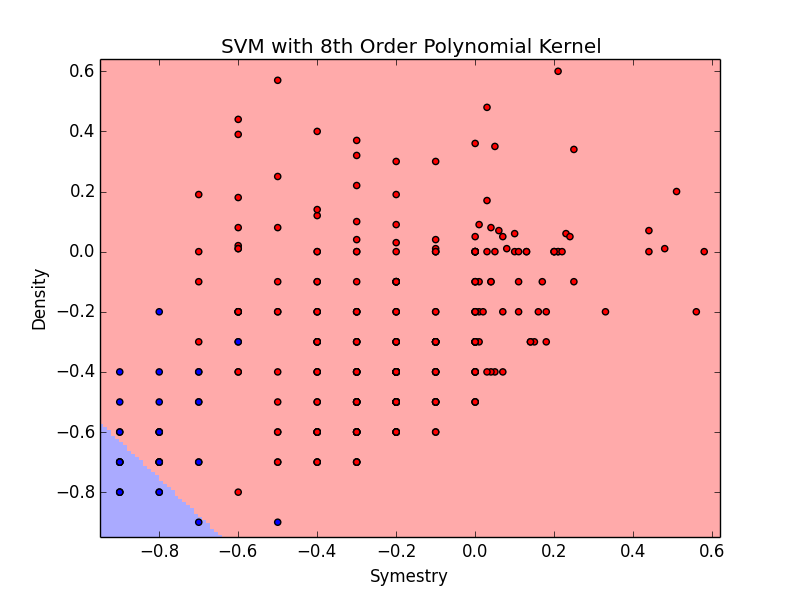
1. SVM

c. The decision boundary for classifier with and without feature transformation is:

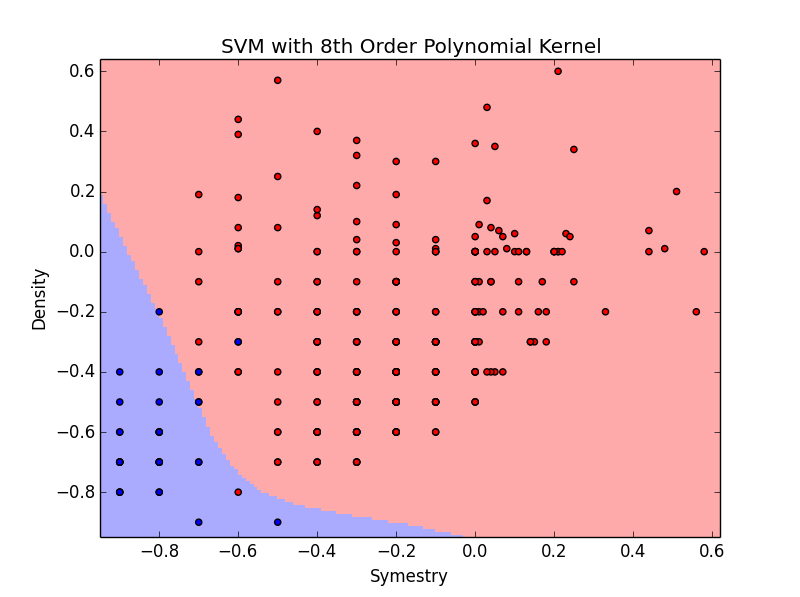


Where the red line is the optimal hyper-plane separator without feature transformation, the blue line plots the decision boundary with feature transformation in Z space.

1. **SVM with Digits Data**
2. I chose the value of C to be 0.01 and 10, and got the classifier corresponding to each value of C as below:



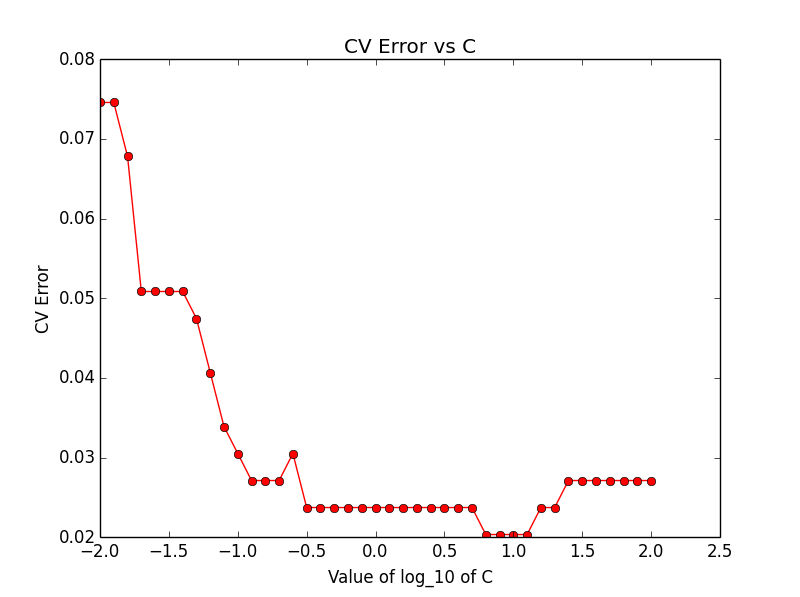
**When C=0.01**

****

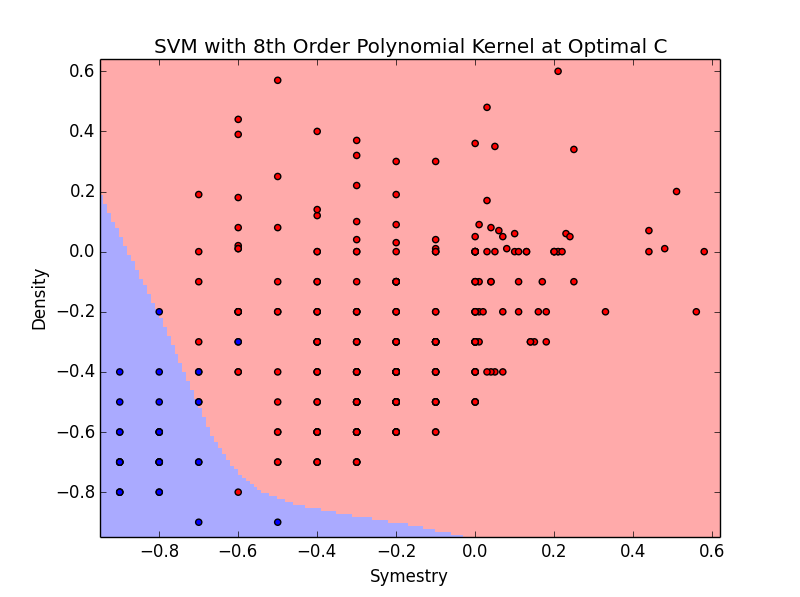
**When C=10**

Smaller value of C may put more regularization on the fitting parameters, thus potentially lead to under-fitting as shown in above figures.

1. The complexity of the decision boundary increases as the value of C increases. It is because where is the regularization parameter. Thus, the larger C is, the smaller is, the less regularization is employed, we are more likely to get a complex decision boundary and more prone to over-fitting.
2. I used CV to find a proper value of C. The CV errors against different choice of C are plotted as below:



I chose C=10 as it minimize the CV error. The resulting classifier is as below:



The testing error for this classifier is 0.04068.



1. **Compare Methods**

Different classification models are compared in terms of the testing error:

LR, KNN, RBF-network, Neural Network and SVM

|  |  |
| --- | --- |
| Approach | Testing Error |
| LR | 0.0893 |
| KNN | 0.0345 |
| RBF Network | 0.0402 |
| Neural Network | 0.0350 |
| SVM | 0.0407 |

* Neural Network, SVM, RBF Network and KNN are more powerful than LR
* The pseudo inverse algorithm simplifies the computation of LR a lot.
* KNN is computational inefficient as distance measures is required.
* SVM would be more flexible but cross validation is required for selecting proper parameters.
* Neural network performs well with proper choice of layers and number of nodes in each layer.
* RBF network also performs well in terms of classification error, but requires determining a proper number of centers K.